REMARKS

In view of the foregoing amendments and following remarks response to the Office Action dated January 29, 2007, Applicant respectfully requests favorable reconsideration of this application.

Applicant respectfully thanks the Office for withdrawing all of the previous rejections.

However, the Office has asserted new rejections against all of the pending claims, i.e., claims 1-48.

Matters of Form

In sections 3-6 of the Office Action, the Office rejected claims 4 and 7 under 35 U.S.C. 112, second paragraph, as being indefinite. Applicant has herein amended the claims to overcome these rejections.

Prior Art Rejections

In sections 7-27, the Office rejected claims 1-4, 7, 11-28, 31, and 35-48 under 35 U.S.C. 102(e) as being anticipated by Barnett. In sections 28-31, the Office rejected claims 5, 6, 8, 9, 29, 30, 32, and 33 under 35 U.S.C. 103(a) as unpatentable over Barnett in view of Sycara. Finally, in sections 32-33, the Office rejected claims 10 and 34 as unpatentable under 35 U.S.C. 103(a) over Barnett in view of Project JXTA.

Applicant respectfully traverses.

The Present Invention

The present invention relates to the maintenance and movement of Web services in a network. As described on page 3, lines 11 et seq. of the present specification, "Web services" are application logic or software modules that can be exposed to and shared with other nodes over the Internet via a standardized interface mechanism.

The invention provides a software construct (termed a Web service "container" in the specification) for managing Web services at a network node and an adaptive model for the dynamic configuration for a plurality of Web service containers distributed throughout a network in a software and hardware platform-independent manner based on contextual information, such as the identity of the Web service, the capabilities of the Web service, the operating system of the Web service, the platform of the Web service, the Web services hosted by a container, the workload of the Web service, and the network location of the Web service. The containers dynamically adapt themselves and, particularly, the Web services contained therein based on a pluggable set of heuristics. The containers can exchange Web services software on a peer-to-peer basis or through querying a registry. The containers can discover the Web services contained in other containers. The present invention allows containers to dynamically exchange Web services software as well as contextual information, such as current workload, so that containers are virtually limitlessly reconfigurable based on context. For instance, containers may, not only reconfigure themselves as routers to other containers containing requested Web services, but also load and unload Web service software modules based on detected workload and/or Web service availability at other servers and send software modules to peer containers as needed to allow them to run a Web service locally rather than from a remote location on the network.

The invention also enables one server with a Web services container to send Web services software to another server with a Web services container in order to allow that other server to begin providing that service. This may be useful, for instance, when the work load at a first server exceeds that server's capabilities. That server can then send the Web service software to one or more other servers and then divide the servicing of requests for that service among two or more servers. The first server can reconfigure itself either partially or totally as a "service router" to route requests for given Web services to other servers that it has determined can provide that service either by virtue of it having itself sent the Web service software to the other server(s) or by querying the other server(s) as to the contents of their Web service containers.

Another exemplary routing scenario involves a request being received by the first server. Acting as a "service router", the first server directs the request to the second server. When the second server responds to the request, it includes context information that indicates it was the node that ultimately handled the request. The container that initiated the request understands the contextual information returned and, for each subsequent request, directs the outgoing message to the second server. This can be logically thought of as dynamically adding a WSDL port to the service and indicating that the new port is the preferred endpoint. In this manner, any number of "hops" may be taken before reaching the ultimate destination, but network efficiency is gained by providing a mechanism that avoids unnecessary processing from the second service request forward.

Not only can Web services software be exchanged in a platform independent manner, but the Web service containers themselves are platform independent. That is, the Web service containers at two different network nodes can be implemented in different programming languages and run on different platforms, while still being able to exchange contextual information and Web service software modules using SOAP and WSDL.

The Barnett Reference

Barnett pertains to a distributed computing platform architecture for facilitating dynamic availability of e-commerce services over a network. Barnett describes a system in which client machines access Web services available at network servers. Discovery by the clients of what Web services are available to it is done by consulting with a particular Web server 200 that maintains a database of the available Web services. Abstract of Barnett.

Particularly, a client that wants to know what Web services are available to it logs on to server 200 and provides the server with its group identification. Server 200 maintains a database indicating what Web services are available to members of the different groups and provides that information to the client. The Web services that are

available to the clients are not on the server 200 but are on different servers. Paragraphs [0031], [0033], and [0035]-[0036]. The client machines, server 200, and the other servers 230 with which the Web services are associated all have different software for interacting with the other types of nodes on the network.

This is a completely different paradigm than that of the present invention. First, Barnett's use of the term Web services refers to a very different thing than the Web services discussed in the present application. Second, Barnett discusses client machines accessing and using Web services found on servers and does not relate to the discovery, exchange and dynamic reconfiguration of Web services between and amongst a plurality of servers. Furthermore, one of the key features of the present invention is that a plurality of servers on a network have the same piece of software, i.e., the "container", for carrying out the discovery and exchange of Web services. Furthermore, the container itself is a Web service.

Barnett pertains to a client/server type of paradigm for discovering and exchanging Web services. For instance, a client having the client software can consult server 200 and LDAP 220 to determine what Web services are available to it. However, Barnett does not disclose server 200 determining what Web services it has locally, or downloading Web services locally, or the other servers 230 determining what Web services are available at other servers. Barnett does not discuss how servers 230 obtain copies of the Web services. Barnett contains no discussion of any contextual information, such as workload, that is exchanged between nodes of the network. Barnett is concerned with a different issue, namely, clients using the Web services found on the servers.

Each of the three different types of nodes on the network discussed in Barnett (clients, server 200, and servers 230) has different software for performing different functions particular to that type of node. This is very different than the present invention which is primarily a peer to peer paradigm in which all of the servers have the same "container" software and manage, disclose, and discover what Web services they themselves have and what Web services are available at other nodes of the network,

and permit exchanging of Web services, serving as proxy nodes for Web services, etc. Thus, Barnett discloses a completely different paradigm than the present invention. No single node performs all of the recited functions of the container of the present invention.

Discussion

The independent claims, claims 1 and 25 clearly recite these distinctions over Barnett. For instance, claim 1 recites that the containers are located at a particular network node and disclose the Web services that are available at that particular node. Claim 1 further recites that the container is itself a Web service.

The Office has asserted that Barnett discloses in paragraph [0037] that its software is a Web service also. However, paragraph [0037] merely discloses that the ExecutionManager software module is a servlet. There are numerous flaws with the Office's analysis in this regard. First of all, the fact that ExecutionManager is a servlet does not make it a Web service. As disclosed in paragraph [0037]:

> [0037] ExecutionManager 260 is a servlet which provides a mechanism for all services to run. In particular, the ExecutionManager 260 listens for communications from the client applet and if it reads a valid Executable, the ExecutionManager passes it to a LoadBalancer/ComputeServer(s) 270. The ExecutionManager 260 also uses the LookupFinder class to find the available LoadBalancers to execute computationally intensive jobs. ExecutionManager also utilizes a ServiceFinder class to find ComputeServers. If there is an error reading the Executable, an error message is returned to the client applet. When the ExecutionManager has read a valid Executable, it attempts to pass it to the LoadBalancer/ComputeServers (270 of FIG. 2).

This does not meet the definition of Web services expressly set forth in claim 1, namely, "executable software modules that can be exchanged between nodes of a network and run at said nodes".

Secondly, there is no single piece of software in Barnett that performs all the functions recited in claim 1 for a single container. Specifically, claim 1 recites that the

container software module (1) determines the Web services available locally, (2) discloses to other nodes the Web services available at the local node, (3) receives and deciphers messages from other nodes to discover what Web services are available at those other network nodes, and (4) can dynamically reconfigure Web services available at its node based on messages exchanged between the nodes.

There is no single piece of software in Barnett that can perform all of these functions because clients 205, server 200, and servers 230 all have different software modules that perform different functions. No single node has software that performs all of the functions recited in claim 1 for the "container".

This is a completely different paradigm. One of the primary points of the present invention is that all of the nodes have the same piece of software that can perform all of the aforementioned functions and that the software itself is a Web service. Barnett teaches a completely different paradigm in which clients, servers 230, and server 200 all perform different functions. Also, nothing in Barnett suggests that the software modules that perform the functions that the Office relies on are Web service software modules themselves.

Finally, Applicant has amended claim 1 to include a recitation that the messages transmitted between containers also include contextual information. As described in the specification, contextual information comprises information about the context of the Web service, such as the capabilities of the Web service, the operating system of the Web service, the platform of the Web service, the Web services hosted by a container, the workload of the Web service, and the network location of the Web service.

Barnett does not disclose the transmission of any contextual information about the services. In Barnett, for instance, a separate LoadBalancer 270 handles such tasks as determining the most eligible Compute Server. Paragraphs [0038]-[0040]. Contextual information in Barnett is handled within the Execution Manager 260 and/or the LoadBalancer 270.

Claim 25 even more clearly distinguishes over the prior art of record. Particularly, claim 25 is a system claim directed to the interaction between the

containers at various nodes on the network. Particularly, claim 25 recites at least the following distinctions over Barnett: (1) "providing a computer program product (hereinafter "container") at each of a plurality of said network nodes"; (2) "determining and describing Web services software modules that are available at a corresponding network node"; (3) "transmitting messages via the network disclosing said Web services software modules that are available at said corresponding node to other network nodes via said network"; (4) the messages "including contextual information about said container and said Web services available at said corresponding, local node, (5) "receiving and deciphering messages from other network nodes disclosing Web services software modules that are available at other network nodes"; (6) "dynamically reconfiguring Web services software modules on said network node transmitted and received based on said messages"; and (7) "wherein said method is embodied in a Web services software module".

It must be understood that the container of the present invention is a software module that manages the life cycle of Web services available at the corresponding network node and these are the types of functions set forth in independent claims 1 and 25 as discussed above. Barnett's Execution manager, on the other hand "is a servelet that allows a constant point of entry for the client applet. The ExecutionManager reads in an executable object from the client applet and passes it to a LoadBalancer. The LoadBalancer is found through a Lookup Server". Barnett, paragraph [0101]. The ExecutionManager does not manage the life cycle of the Web services, rather it is an entry point into the Web services.

Accordingly, independent claims 1 and 25 distinguish over the prior art of record. The dependent claims distinguish over the prior art of record for at least all of the same reasons set forth above with respect to the independent claims. The secondary references cited in connection with the obviousness rejections do not overcome the shortcomings of the primary reference Barnett discussed above. Accordingly, the obviousness rejections also have been overcome.

Additional Distinctions of Dependent Claims

In addition dependent claim 7 further distinguishes over the prior art of record by reciting that "said computer executable instructions for transmitting messages uses a peer to peer messaging protocol between said containers and said computer executable instructions for receiving and deciphering messages uses a peer to peer messaging protocol between containers". Dependent claim 31 contains very similar recitations. The Office asserted that Barnett teaches these recitations, but, unlike the other claim rejections, has not cited a portion of Barnett to support this proposition. Applicant cannot find a portion of Barnett that supports this proposition.

Dependent claims 12 and 36 recite that, responsive to receipt of a client request for a Web service that is not available at the corresponding node, the container determines whether another node has a copy of the particular Web service and invokes a proxy to the container at the other node. The Office asserted that this is found in paragraph [0035] of Barnett. However, paragraph [0035] does not disclose the dynamic reconfiguration of Web services at all. It only discusses the discovery of new Web services (or disappearance of old Web services). It does not discuss any actual dynamic reconfiguration and it certainly does not contain any discussion of invoking proxies to other containers.

Most of the remaining dependent claims recite with specificity some of the dynamic reconfigurations that the container can carry out. Each of these claims clearly adds further distinguishing features over the prior art insofar as Barnett does not perform any dynamic reconfiguration of Web services amongst a plurality of servers (even if one were to consider the claimed Web services to read on Barnett's "Web services", which it does not, as discussed above). Barnett is concerned with the use of services by client machines. The present application does not pertain to the use of Web services (even accepting, arguendo, the Office's broad definition of Web services), but to the management of Web services amongst a plurality of servers.

More specifically with reference to the dependent claims, dependent claims 13 and 37 recite that the proxy routes the client requests for a Web service that is not

available at the corresponding node to the other node and receives the responses from the other node and forwards them to the requesting client. The Office asserted that that this is found also in the same paragraph [0035] of Barnett. However, as discussed above, paragraph [0035], not only does not disclose one container passing requests for a Web service and responses from a Web service to another node, it does not even pertain to the operation of Web services, but only to the discovery of Web services.

Dependent claims 14 and 38 depend from claims 13 and 37, respectively, and recite the steps performed by the container that receives a request from the proxy container. The Office cited paragraph [0049] of Barnett for allegedly teaching this feature. However, paragraph [0049] simply discusses a client machine downloading a Web service from a server. This has nothing to do with proxying.

Claims 16 and 40 recite a dynamic reconfiguration scheme in which, when the load of requests for a local Web service exceeds a predetermined level, the container sends out a request to another container at another node that as a copy of the Web service for a copy of the Web service. The Office asserted that this is found in paragraph [0039] of Barnett. However, paragraph [0039] discusses a LoadBalancer/ComputeServer 270 that passes jobs to the most eligible ComputeServer based on various criteria. However, it does not discuss the loading of the Web services software modules themselves. Paragraph [0039] pertains strictly to load balancing of requests for Web services, not exchanging of Web services between nodes.

Dependent claims at 17 and 41 depend from claims 16 and 40, respectively, and recite the feature of the node offloading the Web service after the load for that Web service drops below a second predetermined level. The Office again cites paragraph [0039] of Barnett as disclosing this feature. However, as noted above, paragraph [0039] has nothing to do with this.

Claims 18,19, 42, and 43 recite a different dynamic reconfiguration and load balancing scheme in which, when the load of requests for a particular Web service at a particular node exceeds a threshold, the container at that node asks the container at another node to accept a copy of the Web service from it, sends a copy of the Web

service to the other node, and then routes requests that it receives for that Web service to the other node. The Office asserts that paragraphs [0037] and [0039] of Barnett disclose this scheme. However, as previously noted, these paragraphs do not discuss rerouting of requests or copying or exchanging of Web services. These paragraphs discuss only load balancing of requests among a plurality of servers that already have copies of the Web service.

Other dependent claims recite additional distinguishing features.

Conclusion

In view of the foregoing amendments and remarks, this application is now in condition for allowance. Applicant respectfully requests the Examiner to issue a Notice of Allowance at the earliest possible date. The Examiner is invited to contact Applicant's undersigned counsel by telephone call in order to further the prosecution of this case in any way.

Respectfully submitted,

Dated: May 16, 2007

/Theodore Naccarella/
Theodore Naccarella
Registration No. 33,023
Synnestvedt & Lechner LLP
2600 Aramark Tower
1101 Market Street
Philadelphia, PA 19107
Telephone: (215) 923-4466
Facsimile: (215) 923-2189

TXN:pmf

S:\\IBM\IBM Raleigh RSW\Patents\P25391 USA\PTO\Response-to-1 29 07 OX v3.doc